Pekka Korhonen

SÄHKÖSÄÄENNUSTESOVELLUKSEN KEHITTÄMINEN

Tekninen toteutus Windows Phone 7 -käyttöjärjestelmälle sekä hintapäivityksen automatisointi

Opinnäytetyö
KESKI-POHJANMAAN AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Tammikuu 2013

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Ylivieskan yksikkö	**Aika** Tammikuu 2013	**Tekijä/tekijät** Pekka Korhonen
Koulutusohjelma Mediatekniikan koulutusohjelma		
Työn nimi SÄHKÖSÄÄENNUSTESOVELLUKSEN KEHITTÄMINEN. Tekninen toteutus Windows Phone 7 - käyttöjärjestelmälle sekä hintapäivityksen automatisointi		
Työn ohjaaja Hannu Puomio		**Sivumäärä** 34 + 2
Työelämäohjaaja Terho Kivimaa		

Tämän opinnäytetyön aiheena oli suunnitella ja toteuttaa pienoisohjelma Windows Phone 7 -käyttöjärjestelmälle. Sähkösääennustesovelluksen tarkoitus oli antaa tietoa loppukäyttäjälle energioiden pidemmän aikavälin hintakehityksestä, ja näyttää kuvaajassa tietokannasta haetut tiedot. Sovellukseen tehtiin erikseen kaksi osiota, joista toinen oli pienasiakkaille eli kotitalouksille ja toinen suurasiakkaille eli yrityksille. Yritysosioon kuului lisäksi sähkön suojausennuste.

Tein myös samalla tutkimusta älypuhelimien käytettävyydestä. Älypuhelimien käyttöjärjestelmien ja sovellusten suunnittelussa otetaan huomioon eri tyyppisiä asioita kuin normaalisti esimerkiksi tietokoneen näytölle suunnitellussa sovelluksessa.

Työn toinen osuus oli toteuttaa hintapäivityksen automatisointi. Ennen tiedot lisättiin tietokantaan käsin, mikä oli työläs tehtävä. Tehtävänäni oli toteuttaa Www-pohjainen sovellus, joka osaa päivittää tiedot automaattisesti järjestelmän tietokantaan.

Asiasanat
Käytettävyys, pienoisohjelma, sähkösääennuste, tietokanta

ABSTRACT

CENTRIA UNIVERSITY OF APPLIED SCIENCES	Date January 2013	Author Pekka Korhonen
Degree programme Media technology		
Name of thesis ENERGY WEATHER FORECAST DEVELOPMENT. Technical implementation for Windows Phone 7 operating system and automation of price update		
Instructor Hannu Puomio		**Pages** 34 + 2
Supervisor Terho Kivimaa		

The subject of this thesis was to design and implement a Widget for Windows Phone 7 operating system. Energy weather forecast application was designed to provide information to the end user a longer-term energy price developments, and display the graph in the retrieved data from the database. Application was made separately for two parts, one of which was a small customers and other for company customers. The company part also included a partition-power protection forecast.

I also did a research of the usability of smart phones. Smartphone operating system and applications are designed taking into account the different types of things as you normally would on your computer screen.

Second portion of the work was to implement a price update automation. Before the data was added in the database by hand, which was a tedious task. My task was to implement a Web-based application that can automatically update the data in the system database.

Key words
Database, energy weather forecast, usability, widget

KÄSITTEIDEN MÄÄRITTELY

HTML = WWW-sivuilla käytettävä sivunkuvauskieli

PHP = WWW-sivujen ohjelmointiin tarkoitettu yleiskäyttöinen ohjelmointikieli

SQL = Ei-proseduraalinen tietokannoissa käytettävä kyselykieli

WP7 = Microsoftin kehittämä älypuhelimien käyttöjärjestelmä

WWW = Maailmanlaajuinen tietoverkko

XAML = Merkkauskieli sovelluskehityksessä

ESIPUHE

Sain tämän opinnäytetyön aiheen tietoliikennetekniikan lehtori Hannu Puomiolta keväällä 2012. Aloitin projektin tekemällä työtä ammattiharjoitteluna, jonka jälkeen jatkoin työtä harjoittelujakson jälkeen opinnäytetyönä. Siitä alkoi pitkä ja raskaskin taival, joka on nyt päätöksessään. Aiheena oli suunnitella ja toteuttaa sähkösääennustesovellus Windows Phone 7 -käyttöjärjestelmälle, jota voidaan käyttää nykyaikaisilla älypuhelimilla.

Tavallaan koko opiskeluaika ammattikorkeakoulussa kiteytyi yhdeksi kokonaisuudeksi opinnäytetyön muodossa. Opin tätä työtä tehdessäni erittäin paljon uusia asioita, ja olen myös erittäin onnellinen siitä, että opinnot ovat nyt päätöksessään.

Haluan kiittää ohjaajaani Hannu Puomiota neuvoista ja ohjeista joita sain opinnäytetyötä tehdessäni. Nöyrät kiitokset myös työelämäohjaajalleni toimitusjohtaja Terho Kivimaalle, joka luotti minuun ja kannusti vaikeina hetkinä.

Sievissä 10.1.2013

Pekka Korhonen

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
ESIPUHE
SISÄLLYS

1 JOHDANTO 1

2 KÄYTETTÄVYYS 3
2.1 Käytettävyyden määritelmä 3
2.2 Käytettävyyden psykologia 4
2.3 Arviointimenetelmät 4
2.4 Mobiililaitteiden erityispiirteet 5

3 KÄYTETYT OHJELMAT JA OHJELMOINTIKIELET 7
3.1 HTML -sivunkuvauskieli 7
3.1.1 Sivun perusrakenne 8
3.1.2 Lomakkeet 9
3.1.3 Taulukot 10
3.2 PHP -skriptikieli 10
3.2.1 Esimääritellyt muuttujat 11
3.2.2 Muuttujat 12
3.2.3 Ehtorakenteet 12
3.2.4 Toistorakenteet 13
3.2.5 Funktiot 14
3.2.6 Merkkijonofunktiot 14
3.2.7 PHP ja tietokantayhteys 14
3.3 SQL -kyselykieli 16
3.4 XAML -merkkauskieli 17
3.5 C#.NET -ohjelmointikieli 18
3.5.1 Ohjelman perusrakenne 19
3.5.2 Vastaavanlaiset menetelmät PHP:n kanssa 19
3.5.3 Listat ja taulukot sekä isolated storage 20

4 SÄHKÖSÄÄENNUSTE 21
4.1 Tietojärjestelmän rakenne 21
4.2 Tietokanta 22
4.3 PHP -palvelu 23
4.4 Pienoisohjelma 25

5 HINTAPÄIVITYKSEN AUTOMATISOINTI 28

5.1 Tiedoston lukeminen **28**
5.2 Tiedon käsittely **29**
5.3 Tiedon lisääminen tietokantaan **30**
5.4 Lisätoiminnot **30**

6 TULOKSET JA POHDINTA **32**

LÄHTEET **33**
LIITTEET

KUVIOT
KUVIO 1. Sähkösääennusteen tietojärjestelmän rakenne 22
KUVIO 2. Hintapäivityksen tietojärjestelmän rakenne 28

TAULUKOT
TAULUKKO 1. Kuvitteellinen tietokannan taulu 23

1 JOHDANTO

SQL- ja relaatiotietokanta- opintojakso oli eräänlainen vedenjakaja ammattikorkeakouluopinnoissani. Suorittaessani sitä ensimmäistä kertaa asenteeni opiskeluun oli pielessä, ja tuloksena oli pyöreä nolla. Toisella yrittämällä oli toinen ääni kellossa, ja onnistuin opintojaksolla erittäin hyvin. Silloin päätin, että haluan tehdä opinnäytetyön, joka liittyy jollakin tavalla tietokantoihin. Tämä toiveeni toteutui keväällä 2012, kun ilmoitin halukkuuteni osallistua Energiaguru -projektiin ammattiharjoitteluna. Alussa en osannut arvioida kunnolla, mihin olin ryhtymässä, mutta jälkeen päin ajateltuna päätös oli erittäin hyvä. Ammattiharjoittelun aikana työtehtäväni kuitenkin muuttui, ja halusin saattaa työn loppuun opinnäytetyönä.

Tutkin työtä tehdessäni erittäin olennaista osaa sovelluskehityksessä eli käytettävyyttä. Käytettävyyden tavoitteena on taata, että sovelluksen käyttäminen on sujuvaa ja että loppukäyttäjä käyttää sovellusta vielä uudemman kerran. Käytettävyyden teoriaan kuuluu muun muassa psykologia ja arviointimenetelmät, joista tein tutkimusta ensimmäisessä pääluvussa. Otin käytettävyystutkimuksessa erityisesti huomioon mobiililaitteiden haasteet.

Toisessa pääluvussa on esitelty käytetyt menetelmät teknisessä toteuttamisessa. Käytännön työssä tarvittiin useita sovelluskehitysohjelmistoja ja työkaluja. Kolmas pääluku käsittelee pienoisohjelman toteuttamista, joka toteutettiin Windows Phone 7 -käyttöjärjestelmälle. Sähkösääennusteen tarkoituksena on antaa asiakkaalle tietoa energioiden hinnoista pitkällä aikavälillä. Sen avulla asiakas voi seurata hintojen kehitystä ja tulevaisuuden näkymiä. Sähkösääennusteessa on esitelty yleisimmät Suomessa käytössä olevat sopimustyypit. Sähkösääennusteesta oli kaksi osiota, joista toinen oli kotitalouksille ja toinen yritysasiakkaille.

Pääluvussa neljä on kuvattu hintapäivityksen automatisoinnin toteuttaminen. Www-pohjaisen sovelluksen tarkoitus on auttaa päivittäjää, sillä aiemmin tiedot jouduttiin lisäämään tietokantaan käsin. Nyt urakka helpottui huomattavasti, kun tietokoneohjelma tekee suurimman osan työstä. Työn tulokset ja muu pohdinta on viimeisessä pääluvussa kuusi.

2 KÄYTETTÄVYYS

Älypuhelimille ja tietokoneille on tullut yhä selkeämmät roolit päivittäisessä käytössä. Älypuhelimet ovat jo joiltakin osin syrjäyttämässä tietokoneet. Tietokonetta käytetään yhä työn tekemiseen ja verkkopankkiasiointiin, mutta monia muita palveluita käytetään usein älypuhelimilla, kuten sosiaalinen media, tiedonhaku ja karttapalvelut. (Adage 2011.)

Siksi on tärkeää, että varsinkin älypuhelimissa suoritettavien sovellusten käytettävyys olisi mahdollisimman sujuvaa ja käyttökokemukset olisivat positiivisia. Yhä useampi tavallinenkin kansalainen omistaa älypuhelimen, ja lisääntyvien ominaisuuksien myötä sitä käytetään myös moneen muuhun asiaan kuin pelkästään soittamiseen ja tekstiviesteihin.

2.1 Käytettävyyden määritelmä

Käytettävyys on menetelmä- ja teoriakenttä, jonka kautta käyttäjän ja laitteen yhteistoimintaa pyritään saamaan tehokkaammaksi ja käyttäjän kannalta miellyttävämmäksi. Käytettävyys käyttää hyväksi kognitiivisen psykologian sekä ihmisen ja koneen vuorovaikutuksen tutkimusta. Hyvän käytettävyyden muodostavat käyttötilanteen opittavuus, virheettömyys, muistettavuus, tehokkuus ja miellyttävyys. (Sinkkonen, Kuoppala, Parkkinen & Vastamäki 2006, 17.)

Voidaan todeta, että sovelluksen tai laitteen käytettävyys on erinomainen silloin, kun loppukäyttäjä oivaltaa asioita käyttäessään ja oppii nopeasti sen, mitä suunnittelija on jollakin tietyllä ominaisuudella hakenut. Käytettävyys on hyvä

silloin, kun loppukäyttäjä ei erityisesti kiinnitä siihen huomiota. Käytettävyyden suunnittelussa on epäonnistuttu silloin, kun loppukäyttäjä kokee laitteen tai sovelluksen käytön vaikeaksi ja epämiellyttäväksi.

2.2 Käytettävyyden psykologia

Jokaisella käyttäjällä on seuraavia ominaisuuksia: oppiminen, muistaminen, tarkkaavaisuus ja motivaatio. Ongelma on aikaansaada ohjelma, joka noudattaa edellä mainittuja ominaisuuksia. Suunnittelijoiden pitää tietää asioita, jotka koskevat käyttäjän ja tuotteen välistä vuorovaikutusta, miten asioiden laita oikeasti on. (Hangasmaa 2010.)

Ihminen oppii palvelun tai tuotteen käytön melko nopeasti. Ihminen muistaa pian ulkoa sen, mistä tapahtuu mitäkin ja minne esimerkiksi navigointilinkeillä siirrytään. Tarkkaavaisuus tarkoittaa, että ihminen pystyy käsittelemään vain rajallisen määrän tietoa kerrallaan. Motivaatioon vaikuttaa ihmisen sen hetkinen mielentila. Huonona päivänä ihminen turhautuu helpommin ja motivaatio on alhainen.

2.3 Arviointimenetelmät

Tuotteen käytettävyyden rakentamiseksi ja varmistamiseksi on olemassa suuri joukko menetelmiä. Näistä menetelmistä keskeisimmät ovat käyttäjien tehtävien, osaamisen ja toimintaympäristön selvittäminen ja dokumentointi sekä käytettävyystestaus. Käytettävyystestin perimmäinen tarkoitus on tehdä tuotteen käyttölaadusta parempi seuraamalla käyttäjän mentaalimalleja tilanteessa, joka muistuttaa aitoa tilannetta - tai ainakin tehtävät ovat niin aitoja kuin mahdollista.

Käytettävyystestejä on periaatteessa kahden tyyppisiä: kehitystestien tarkoituksena on käytettävyydeltään mahdollisimman hyvän käyttöliittymän löytäminen. Hyväksymistestissä varmistetaan, että tuote täyttää sille asetetut käytettävyysvaatimukset. (Sinkkonen ym. 2006, 275–276.)

Perusperiaatteena siis on, että paras tapa arvioida käytettävyyttä on testauttaa sovellusta sekä keskeneräisenä että viimeistelyvaiheessa muilla käyttäjillä. Ohjelmoijan työssä ei välttämättä aina huomaa sellaisia asioita, mitä tavallinen käyttäjä havaitsee tai haluaa. Joskus myös tekniset rajoitteet luovat tiettyjä rajoituksia myös käytettävyyden ja käyttöliittymän suunnittelussa. Käytettävyydessä ei kuitenkaan voida joustaa liikaa teknisten ratkaisujen takia.

2.4 Mobiililaitteiden erityispiirteet

Älypuhelimissa ja kämmentietokoneissa suurin ja huomattavin ero PC-käyttöön on näytön ja näppäinten pieni koko. Siksi sovelluksien käyttöliittymän toteuttaminen on haasteellista, koska näyttöön ei yksinkertaisesti mahdu paljoa esitettävää tietoa. Haaste on myös siinä, mitä tietoa näytössä esitetään ja millä tavalla. WP7 -kehitysympäristö sisältää muun muassa mahdollisuuden luoda viesti-ikkunan, joka vastaa Internet- ja PC-sovellusmaailmasta tuttuja ponnahdusikkunoita. Mielestäni WP7 -käyttöjärjestelmän suunnittelussa on onnistuttu ihan hyvin. WP7:n aloitusnäkymä koostuu elävistä laatikoista (*live tiles*), joita napsauttamalla haluttu sovellus käynnistetään. Aloitusnäkymästä näkee välittömästi esimerkiksi saapuneet puhelut, tekstiviestit ja sähköpostit.

Jos lasketaan päivittäisen käytön määrä yhteen, käytämme pieniä käyttöliittymiä varmasti enemmän kuin normaalikokoisia tietokoneita. Pienten käyttöliittymien suunnittelu on siis nykyajan haaste varsinkin älypuhelimien ja pienten

multimediatietokoneiden yleistyessä. Pienissä käyttöliittymässä vuorovaikutuksen suoraviivaisuus, yksinkertaisuus ja tehokkuus nousevat hyvin tärkeiksi asioiksi. (Leander 2006.)

3 KÄYTETYT OHJELMAT JA OHJELMOINTIKIELET

Opinnäytetyö koostui käytännönläheisesti kahdesta osakokonaisuudesta eli mobiilipienoisohjelmasta sekä hintapäivityksen automatisoinnista. Ne toteutettiin tarkoituksiin soveltuvilla ohjelmistoilla ja ohjelmointikielillä. Pienoisohjelman sovelluskehitystyökaluna oli Microsoftin Visual Studio 2010 Windows Phone lisäosan kera. Ohjelmointi toteutettiin C#.NET -ohjelmointikielellä. Hintapäivitysohjelma toteutettiin Adoben Dreamweaverilla PHP -ohjelmointikielellä.

3.1 HTML -sivunkuvauskieli

Periaatteessa hintapäivityksen olisi voinut toteuttaa täysin ilman suoraan toteutettua HTML -kieltä. On kuitenkin hyvän tavan mukaista, että kaikki mahdollinen staattinen sisältö toteutetaan HTML -kielellä ja dynaamisuus PHP -kielellä. Tämä siksi, että silloin vältytään pitkiltä ja hankalan näköisiltä PHP -tulostuslauseilta, joihin on sisällytetty normaalia HTML -koodia.

HTML on eräs rakenteinen dokumenttien muoto, joka on käsiteltävissä erilaisissa tilanteissa, kuten näytettävissä isolla tai pienellä kuvaruudulla ja tulostettaessa paperille. Sana rakenteinen tarkoittaa sitä, että HTML-dokumentti sisältää tekstisisällön lisäksi sen loogisen rakenteen osoittavaa merkkausta (*markup*). Tavallisesti HTML:ää käytetään webissä, jota varten se on kehitettykin. Sitä voidaan käyttää myös dokumenttien yleisenä jakelumuotona. HTML:ää kutsutaan usein kieleksi, tarkemmin sanoen merkkauskieleksi. Nimi HTML johtuukin sanoista hypertext markup language (hypertekstin merkkauskieli). (Korpela & Linjama 2005, 70.)

3.1.1 HTML -sivun perusrakenne

HTML -sivu alkaa HTML -merkillä (*tag*). HTML -elementtiin voidaan sisällyttää esimerkiksi kielimäärityksiä ja muita sivustoon liittyviä määrityksiä. On syytä muistaa, että HTML -kielellä luodut elementit täytyy muistaa lähes poikkeuksetta sulkea. Sulkeminen tapahtuu lisäämällä merkin sisälle alkuun kauttaviiva. HTML -merkin jälkeen voidaan asettaa HEAD elementti, johon voidaan sisällyttää sivustolla käytettävät tyylitiedostot ja muutakin metatietoja, kuten käytettävä merkistökoodaus. HEAD -osion sisään voidaan määrittää ulkoinen otsikko TITLE, joka tarkoittaa sivun otsikkoa. Otsikko näytetään Www-selaimen yläpalkissa.

```
<html>
<head>
<title>Sivun otsikko</title>
</head>
<body>
<h5>Tämä on h5-otsikko</h5>
<p>Tämä on kappale HTML-sivulla.</p>
Tämä on tekstiä ilman erityistä määrittelyä. Rivinvaihto
voidaan tehdä <br /> tagilla. Teksti voidaan myös
<b>lihavoida.</b>
</body>
</html>
```

Ylläoleva koodi on esimerkki erittäin yksinkertaisesta Www-sivun pohjasta. Kuitenkin HTML -editorit usein lisäävät sivuihin omia määrityksiä. Vaikka määritykset voivat olla hankalan näköisiä, niitä usein tarvitaan kun halutaan vahvistaa (*validate*) sivuston oikeellisuus. Periaatteessa validointi-sana tarkoittaa dokumentin tarkistamista ohjelmalla, joka tutkii, onko sivu HTML-kielen

määrittelyn mukainen (Korpela & Linjama 2005, 155). Internetissä on ilmaisia validaattoreita juuri tätä tarkoitusta varten. Nykyisin eräs tärkeä määritys sivun alussa on DOCTYPE. Sen perusteella Www-selain saa tiedon siitä, miten luotu sivu on tarkoitus näyttää.

3.1.2 Lomakkeet

HTML -lomakkeet ovat oiva tapa helpottaa sekä loppukäyttäjän käyttökokemusta että ohjelmoijan työtä sivustoa suunnitellessa. Lomakkeet lisäävät sivustolle vuorovaikutusta, sillä lomakkeiden avulla käyttäjä voi tehdä esimerkiksi erilaisia valintoja, syöttää omaa tekstiä ja lähettää lopuksi lomakkeen painonappia painamalla. Ohjelmoija puolestaan määrittelee valintojen ja syötteiden perusteella toiminnallisuuden sivulle. Hyvin usein lomake prosessoidaan eli käsitellään erillisellä PHP -tiedostolla. Alun perin lomakkeita luotiin palautelomakkeina, joiden tiedot voitiin lähettää esimerkiksi sähköpostiin tai tallentaa tiedostoon.

Tekstikenttä on ehkä yleisin lomakkeen komponentti. Siihen käyttäjä voi kirjoittaa tekstiä yhden rivin verran. Tekstikentälle voidaan määrittää muun muassa maksimipituus. Tekstialue puolestaan on isompi tekstikenttä, johon voi kirjoittaa useita rivejä tekstiä. Salasanakenttä on tarkoitettu salasanoja varten, ja siinä kirjoitettu teksti esitetään asteriksi -merkkeinä. Valintanappia tarvitaan tilanteeseen, jos käyttäjälle annetaan mahdollisuus valita yksi useasta vaihtoehdosta. Silloin on tärkeää muistaa määritellä kaikki valintanapit samalla nimellä asettamalla valintanapin *name* -attribuutti samaksi. Siten saadaan aikaan valintanappiryhmä.

Valintaruutu on rasti ruutuun -kenttä, jonka käyttäjä voi valita tai perua valinnan klikkaamalla sitä uudestaan. Valintavaihtoehtoja voi valita useita.

Tiedostokentällä palvelimelle saadaan ladattua käyttäjän valitsema tiedosto. Tiedosto voidaan valita normaalisti omalta tietokoneelta tai vaikka ulkoiselta kiintolevyasemalta. Lomakkeen toiminnan kannalta tärkein komponentti on lähetysnappi (*submit*). Sen kanssa toimii myös lomakkeen tyhjennys (*reset*). Lähetysnapilla suoritetaan ohjelmoijan laatima toiminto.

3.1.3 Taulukot

Taulukot ovat perinteinen ja melko yksinkertainen tapa esittää tietoa HTML-sivulla. Taulukoita voidaan käyttää hyödyksi esimerkiksi tekstin tasauksessa lomakkeella, mutta taulukoiden käyttö on pikku hiljaa väistymässä ja tilalle on tullut muita menetelmiä. Yksi uusista menetelmistä on lohkorakenne (*div*). Niiden ulkoasun käsittely CSS -tyylitiedostolla on vaivattomampaa. Seuraava esimerkki kuvastaa yksinkertaisen taulukon luonti HTML-sivulla.

```
<table id="tuotetaulukko" border="1">
<tr><th>Nimi</td><th>Hinta</th></tr>
<tr><td>Porkkana</td><td>2 €</td></tr>
<tr><td>Peruna</td><td>3 €</td></tr>
</table>
```

Edellä luodaan taulukko jossa on kaksi riviä ja saraketta. Rivi alkaa <tr> merkinnällä ja solu puolestaan <td> merkinnällä. Otsikkosarake tehdään <th> merkinnällä.

3.2 PHP -skriptikieli

PHP (HyperText Preprocessor) on skriptikieli, joka on tarkoitettu palvelinpuolen ohjelmointiin. Www-palvelimelle asennettava PHP -ohjelmisto tulkkaa PHP -ohjelman. Ohjelman generoima tulos palautuu asiakkaalle (yleensä www-selaimelle) esitettäväksi. PHP:n suosio on kasvanut viime vuosina vauhdilla ja samalla sen laajuus ja ominaisuudet ovat lisääntyneet. PHP-kieli on korkean tason kieli niin kuin useimmat muutkin nykyajan ohjelmointikielet. Käsite 'korkea taso' viittaa siihen, että ohjelmointi tapahtuu käsittein, jotka ovat ihmiselle tuttuja ja luonteenomaisia. (Kolehmainen 2006, 3.)

PHP-kieli on tärkeä osa Www-ohjelmointia, koska sillä saadaan sivustolle dynaamisuutta. PHP voidaan upottaa suoraan HTML -koodin sekaan (The PHP Group 2012a). PHP ei vaadi tukea loppukäyttäjän Internet-selaimelta, mutta palvelimessa on tietysti oltava tuki PHP:lle. Muuten sivusto ei toimi oikein. PHP on siis omiaan dynaamisille sivustoille, joissa tietoa päivitetään. Hyvin harvoin halutaan tehdä sellainen sivusto, jonka sisältö pysyy täysin koskemattomana (Juola 2011).

3.2.1 Esimääritellyt muuttujat

PHP sisältää muutamia valmiiksi määriteltyjä muuttujia. Näistä yleisimmät ovat $_GET ja $_POST. Näillä esimääritellyillä muuttujilla tieto saadaan vastaanotettua esimerkiksi lomakkeelta, joita käsittelin HTML-osiossa. Näiden muuttujien avulla tieto on käytettävissä PHP -ohjelmassa sellaisenaan. Myös tiedoston lataamiseen on olemassa $_FILE -muuttuja. Lisäksi on olemassa $_SESSION muuttuja, jonka avulla voidaan luoda istunto. Tällöin tieto on käytössä aina, kun istuntomuuttuja on ensin määritelty ja tämän jälkeen istunto on aloitettu.

3.2.2 Muuttujat

PHP:ssa muuttujat alkavat aina dollarimerkillä $. PHP on niin sanotusti heikosti tyypitetty kieli mikä tarkoittaa, että muuttujalle ei aseteta erikseen tietotyyppiä. Se voi olla tilanteesta riippuen esimerkiksi merkkijono, kokonaisluku, liukuluku tai totuusarvo. PHP on siis tietyllä tavalla helppo ohjelmointikieli muuttujien osalta, mutta joskus tietotyypeillä voitaisiin välttää joitakin virheitä.

```
$merkkijono = 'Tämä on tekstiä';
$kokonaisluku = 23;
$totuusarvo = false;
```

Yllä oleva koodi on esimerkki muuttujien käytöstä PHP-kielessä. PHP:ssä on myös mahdollista tehdä useiden muiden ohjelmointikielien tavoin vakioita. Ne tunnistaa sanasta *define*. PHP:ssa on mahdollista myös luoda taulukoita. Ne helpottavat usein tiedon tallentamista ja käsittelyä. Taulukko voi olla moniulotteinen. Taulukko voidaan toteuttaa avain-arvopari -menetelmällä tai ilman niitä. Seuraava esimerkki kuvastaa taulukon luontia:

```
$tuotteet = array(
array('tuotenumero'=>1, 'nimi'=>'Porkkana'),
array('tuotenumero'=>2, 'nimi'=>'Peruna'));
```

Edellä luodaan moniulotteinen taulukko. Taulukko $tuotteet pitää sisällään kaksi taulukkoa, jotka ovat tässä tapauksessa tuotteita. Niille on määritelty avain-arvopari -rakenne. Ensimmäinen avain on tuotenumero, ja sen arvo on porkkanalla yksi. Toisen parin avain on nimi, ja arvona on siis porkkana.

3.2.3 Ehtorakenteet

Hyvin usein ohjelmoinnin opettelu aloitetaan ehtorakenteista. Niillä koodi voidaan määrätä suorittamaan tietty ohjelman osa riippuen esimerkiksi käyttäjän valinnoista. Ehkä yksinkertaisin ehtorakenne on if-else -rakenne. Yksinkertaisimmillaan jos joku ehto on tosi, suoritetaan ensimmäinen toiminto – muulloin suoritetaan toinen toiminto. Joskus jos ehtoja on paljon, on suotavampaa käyttää monivalintarakennetta (*switch-case*). Silloin jokainen tilanne määritellään yhtenä tapauksena. Samaan tapaukseen voi sisällyttää useita vaihtoehtoja.

3.2.4 Toistorakenteet

Jos ohjelman suorituksessa esiintyy samankaltaisia säännöllisiä tilanteita, toistorakenteet helpottavat ohjelman kehittämistä huimasti. For -silmukalla voidaan esimerkiksi käydä läpi taulukko askel kerrallaan. While -rakenne on samankaltainen kuin for, mutta se määritellään hieman eri tavalla. Foreach sopii tilanteisiin, jossa ei vaadita säännöllisiä avain-arvo -pareja.

```
for ($i = 1; $i <= 10; $i++)
{
          echo "Luku on nyt " . $i . "<br>";
}
```

Edellä esitetty PHP -koodi tulostaa allekkain luvut yhdestä kymmeneen. Lukujen eteen tulostetaan myös tekstiä. Oleellista toistorakenteissa on se, miten se on määritelty kulkemaan. Edellä olevassa esimerkissä silmukka alkaa luvusta yksi, ja sitä tulostetaan niin kauan kuin luku on pienempi tai yhtä suuri kuin kymmenen. Jokaisella kerralla lukua kasvatetaan siis yhdellä. Kulkeminen voidaan myös toteuttaa niin, että se alkaa luvusta kymmenen ja pienenee jokaisella kierroksella yhdellä. Lisäksi muuttuja voidaan määritellä kulkemaan esimerkiksi kaksi

pykälää kerrallaan, tällöin edellä kuvatun esimerkin $i -muuttujan kasvatus tulisi muotoon $i += 2.

3.2.5 Funktiot

Funktioita tarvitaan selkeyttämään ja helpottamaan ohjelman suunnittelua. Oma funktio on helppo tehdä ja se on helposti kutsuttavissa muualla koodissa. Jos esimerkiksi ohjelmassa täytyy monessa paikassa suorittaa hinnan alennuksen lasku, funktioon kirjoitetaan alennuskaava. Sille voidaan antaa parametreiksi alkuperäinen hinta ja alennusprosentti.

Tässä tapauksessa funktio kannattaa määrittää palauttamaan saatu arvo (*return*). On myös hyvä miettiä tilanne, jos alennusprosentti on nolla. Tällöin voidaan ehtorakenteella määrittää funktio palauttamaan alkuperäisen hinnan, jos alennusprosentti on nolla tai tyhjä. Funktiota voidaan kutsua yksinkertaisesti sen nimellä, ja sulkuihin annetaan parametrit oikeassa järjestyksessä.

3.2.6 Merkkijonofunktiot

Usein voi olla tarvetta käsitellä merkkijonoja. PHP sisältää useita merkkijonon käsittelyyn tarkoitettuja funktioita. Strpos -funktion avulla voidaan etsiä merkkijonosta haluttua merkkijonoa. Nimensä mukaisesti sillä voidaan etsiä hakusanan paikkaa merkkijonossa, mutta se soveltuu hyvin if -rakenteessa tarkoitukseen, jossa haetaan muuttujasta haluttua merkkijonoa. Se tarvitsee parametreikseen mistä etsitään ja mitä etsitään.

```
$sisalto = 'abc';
$haku    = 'a';
$paikka = strpos($sisalto, $haku);
```

Edellä kuvattu ohjelmakoodi palauttaa muuttujaan 'paikka' arvon 0, koska a-merkki löytyy merkkijonon abc aloituskohdasta. Indeksi alkaa nollasta, joten siksi palautetaan arvo 0, eikä 1. Merkkijonofunktioita käytettäessä on tärkeää, että sekä luettavassa tiedostossa että tiedostoa käsittelevässä ohjelmassa on valittu molempiin sama merkistökoodaus. Jos näin ei ole, funktiot saattavat tehdä odottamattomia asioita.

Merkkijonon korvaamiseen toisella merkkijonolla on myös olemassa funktio. Se on nimeltään str_replace() ja sitä käytetään hyvin samankaltaisesti kuin edellä kuvattua strpos -funktiota. Parametreiksi annetaan ensin se mitä merkkijonoa korvataan, ja toisena parametrina millä korvataan. PHP:n kanssa on syytä muistaa, että merkkijonoja ei tule vertailla toisiinsa tavallisilla vertailuoperaattoreilla vaan siihen tarkoitukseen kannattaa käyttää strcmp -funktiota (Kollanus 2012).

3.2.7 PHP ja tietokantayhteys

PHP:lla voidaan melko helposti luoda tietokantayhteys, jonka jälkeen kannasta voidaan hakea tietoa, lisätä, poistaa tai päivittää. Yhteys luodaan mysql_connect -toiminnolla, johon annetaan patametreiksi kannan osoite, käyttäjätunnus ja salasana. Tämän jälkeen voidaan valita käytettävä kanta. Lisäksi voidaan asettaa mitä merkistökoodausta kanta käyttää. Tällöin vältytään merkistövirheiltä. Testausvaiheessa on hyvä antaa yhteyden näyttää virheilmoitukset mysql_error -toiminnolla.

Nykyisin ei enää aina käytetä PHP:n perinteistä mysql_connect -yhteydenluomismenetelmää, vaan sen tilalle on kehitelty muita tapoja. Yksi näistä on PHP Data Objects (PDO). PDO on tietokantaohjelmointimalli, jonka avulla voidaan tehdä helpommin monimutkaisia ja turvallisia kyselyjä. PDO on riippumaton siitä mitä tietokantaohjelmistoa käytetään, kyselyt suoritetaan samalla tavalla (The PHP Group 2012b).

3.3 SQL -kyselykieli

Structured Query Language (SQL) on nimensä mukaisesti strukturoitu kyselykieli. Se ei ole siis mikään ohjelmointikieli vaan niin sanotusti ei-proseduraalinen kyselykieli. SQL perustuu englanninkielisiin avainsanoihin ja se käsittelee tietoa joukkona. Se kattaa muun muassa tietokannan rakenteen määrittelyn ja muokkauksen, kyselyt, muutokset, päivitykset ja poistot. SQL-komentoja voidaan upottaa ohjelmointikieleen sovelluskehityksessä ja SQL on ehkä ainut kieli, jota kaikki suuret valmistajat eli IBM, Microsoft ja Oracle yksimielisesti tukevat. (Luimula 2011.)

SQL jakaantuu käskyjensä perusteella kahteen osaan. Tiedon määrittelykäskyillä voidaan luoda ja poistaa tietokannasta tauluja, tehdä näkymiä ja muokata olemassa olevaa taulua. Seuraava esimerkki havainnollistaa, kuinka MySQL tietokantaan luodaan taulu:

```
CREATE TABLE henkilo (
hetu VARCHAR (11) PRIMARY KEY NOT NULL,
etunimi VARCHAR (30),
sukunimi VARCHAR (50),
titteli VARCHAR (50))
```

Esimerkissä luodaan taulu nimeltään henkilo. Taulun pääavain on hetu ja se ei voi olla tyhjä. Seuraavaksi luodaan etunimi, sukunimi ja titteli -kentät, joiden tietotyyppi on vaihtuva merkkijono ja maksimipituus on suluissa oleva numero. Kun taulu on luotu, voidaan tauluun kohdistaa käyttökäskyjä. Yleisimmät käyttökäskyt ovat INSERT ja DELETE. Kun tauluun halutaan lisätä tietoa, käytetään INSERT INTO -käskyä.

```
INSERT INTO henkilo (hetu, etunimi, sukunimi, titteli)
VALUES ('000000-0000', 'Olli', 'Opettaja', 'Lehtori')

SELECT * FROM henkilo WHERE etunimi = 'Olli'
```

Edellä kuvatuissa käskyissä tauluun henkilo lisätään yksi tietue. Ensimmäisten sulkujen sisällä on kentät, joihin tieto lisätään. VALUES -määrityksen jälkeen on kerrottu pilkuilla eroteltuna ne arvot, jotka lisätään tietokantaan. Kun tieto on lisätty, se voidaan hakea kannasta SELECT -käskyllä. SELECT -lauseessa haetaan ne henkilöt, joiden etunimi on 'Olli'. Tässä tapauksessa kysely palauttaa tuon yhden ainoan rivin, joka kantaan aiemmin lisättiin. Huomion arvoinen toiminto INSERT INTO -lausekkeessa on AUTO_INCREMENT. Sillä saadaan aikaiseksi automaattinen luvun kasvatus aina, kun uusi tietue lisätään kantaan. Hyvin usein taulu identifioidaan automaattisesti kasvavalla id-sarakkeella, jolloin id-kenttää ei tarvitse erikseen määritellä INSERT INTO -lauseessa.

Muut yleisimmät käyttökäskyt ovat DELETE ja UPDATE. DELETE -komennolla taulusta voidaan poistaa tietoa. Päivittäminen tapahtuu UPDATE -komennolla. Päivittäminen on järkevää silloin, kun olemassa olevan tietueen yhden kentän arvo muuttuu. Lisäksi on olemassa monia muita hyödyllisiä komentoja, joilla tietoa voidaan hakea tietokannasta monin eri perustein. MySQL ja muutkin tietokantaohjelmistot osaavat esimerkiksi hyödyntää päivämääriä (DATE) tehokkaasti.

3.4 XAML -merkkauskieli

Extensible Application Markup Language (XAML) on merkkauskieli sovelluskehityksessä. Silverlight -ohjelma on yleensä sekoitus koodia ja XAML -merkkauskieltä. Useimmiten XAML:ia voidaan käyttää ulkoasun visuaalisen ilmeen määrittelyyn, ja koodilla luodaan tapahtumien käsittely mukaan lukien käyttäjän syötteet. Vaikka XAML:a kutsutaan joskus deklaratiiviseksi kieleksi, se ei todellakaan ole täydellinen ohjelmointikieli. XAML voidaan tehdä joko Visual Studion designer-tilassa tai vaihtoehtoisesti Expression Blend -sovelluksella. Yleensä designer vastaa muutoksiin nopeasti ja tarkasti. Vasta monimutkaisemmissa sovelluksissa vaaditaan ohjelman kääntäminen ja ajo. (Petzold 2010.)

XAML ja C#.NET ovat ainakin joiltakin osin verrattavissa aiemmin käsittelemääni PHP:hen ja HTML:ään. Staattinen ulkoasun sisältö on järkevintä tehdä WP-kehityksessä XAML -kielellä ja toiminnallisuus C#.NET:llä tai VB.NET:llä. XAML mahdollistaa muun muassa eri komponenttien - kuten tekstikenttä, radionappi, luetteloruutu ja kolmannen osapuolen komponentit - lisäämisen helposti sovellukseen. Myös ohjelman suunta (*orientation*) on helppo asettaa XAML -koodissa.

3.5 C#.NET -ohjelmointikieli

C# on Microsoftin kehittämä ohjelmointikieli, jolla voidaan rakentaa erilaisia sovelluksia jotka käyttävät .NET ohjelmointikomponenttikirjastoa. C# on tekijöidensä mukaan yksinkertainen, tehokas, tyyppiturvallinen ja myös olio-

orientoitunut. Monet uudistukset C#:ssa mahdollistavat nopean sovelluskehityksen säilyttäen kuitenkin ilmaisuvoimansa ja tyylikkyytensä puhuttaessa C-tyylin kielistä. (Microsoft 2012.)

3.5.1 Ohjelman perusrakenne

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world");
            Console.ReadKey();
        }
    }
}
```

Edellä kuvattu ohjelmakoodi on yksinkertainen C#:lla luotu konsoliohjelma, joka tulostaa konsoliin "Hello world". '*Using*' -sanalla otetaan käyttöön halutut kirjastot. '*Namespace*' määrittelee ohjelman nimiavaruuden. *'Class Program*' määrittelee '*Program*' -luokan. Main -metodi on ohjelman aloituspiste. (Kolari 2012a.) C# -kielellä luotu Silverlight for Windows Phone -projekti on perusrakenteeltaan samanlainen, vaikka esimerkissä kuvattiin yksinkertainen konsoliohjelma.

3.5.2 Vastaavanlaiset menetelmät PHP:n kanssa

Useat PHP -osiossa esittämäni menetelmät ovat C#:ssa hyvin samankaltaisia. Toiminta on täysin toisiaan vastaavaa, vain syntaksi eroaa kuten kaikkia eri ohjelmointikieliä vertaillessa. Näitä menetelmiä ovat muun muassa ehtorakenne

(*if-else*), monivalintarakenne (*switch-case*) ja toistorakenteet (*for, foreach*). Siksi on tarpeetonta käydä samat menetelmät uudestaan läpi. Muistettakoon kuitenkin, että C#:ssa muuttujille annetaan aina tietotyyppi. Muuttujia käsitellessä tietotyyppi täytyy olla sama. Esimerkiksi merkkijonon ja kokonaisluvun yhteenlasku ei ole mahdollista ilman, että ne ovat samaa tietotyyppiä. C#:ssa on olemassa muunnostyökalu (*convert*) jolla muuttujien tietotyyppi voidaan muuttaa. Seuraava ohjelmakoodi on esimerkki muuttujien arvon asettamisesta C# -ohjelmassa.

```
string merkkijono = "Tämä on tekstiä";
int kokonaisluku = 23;
bool totuusarvo = false;
```

3.5.3 Listat ja taulukot sekä isolated storage

Taulukko voi olla yksiulotteinen, moniulotteinen tai niin sanottu 'jagged array'. Se tarkoittaa, että taulukon yksi alkio on kokonainen taulukko. Alkiolla tarkoitetaan taulukon yhtä yksittäistä arvoa ja siihen voidaan viitata indeksillä. Taulukkoa alustaessa sille annetaan normaaliin tapaan tietotyyppi. Sana jagged viittaa siihen, että alkiot voivat olla myös eri kokoisia. Listat ovat peruskokoelma ilman erityispiirteitä. Siinä on perustoiminnallisuus lisäämiseen, poistamiseen, järjestämiseen ja etsimiseen (Kolari 2012b).

Isolated storage on WP7 -erityispiirre, joka mahdollistaa ohjelman tilan tallentamisen puhelimen muistiin. Tallennus voidaan tehdä esimerkiksi tekstitiedostoon. Tallennettu tieto voidaan myöhemmin lukea muistista ilman, että tieto katoaa esimerkiksi puhelimen soidessa samaan aikaan kun sovellusta käytetään.

4 SÄHKÖSÄÄENNUSTE

Opinnäytetyöni ensimmäinen osuus oli toteuttaa sähkösääennustesovellus mobiilipienoisohjelmana Windows Phone 7 käyttöjärjestelmälle. Pienoisohjelma tarkoittaa sitä, että sovellus on pieni, yksinkertainen ja helppokäyttöinen (Sanastokeskus TSK ry 2012). Pienoisohjelmille on ominaista, että ne näyttävät sovelluksen oleellisimman tiedon nopeasti ja yksinkertaisesti (Qvik Oy 2011). Pienoisohjelmasta voidaan käyttää myös nimityksiä vimpain ja englanniksi widget.

Tavoitteena oli toteuttaa sähkösääennustesovellus pienoisohjelmana, joka näyttää kuvaajassaan Suomessa yleisimmin käytetyt sähkösopimustyypit. Ne ovat toimitusvelvollinen, yhden vuoden määräaikainen tarjous, kahden vuoden määräaikainen tarjous, toistaiseksi voimassaoleva tarjous sekä pörssisopimus. Sähkösääennuste on täysin ilmainen palvelu, ja WWW-sivuilla oleva ennuste kertoo käyttäjälle myös pääsyyt hintojen nousuihin ja laskuihin ja antaa suosituksen nykymarkkinatilanteen parhaasta sopimuksesta.

4.1 Tietojärjestelmän rakenne

Järjestelmä koostuu kolmesta osasta. Kaiken perustana on tietokanta, jonka tietoja voidaan käyttää hyväksi. Järjestelmän keskellä on PHP -palvelu, joka huolehtii tiedon hakemisesta tietokannasta. Viimeisenä on pienoisohjelma, josta voidaan tarkastella kannasta haettuja tietoja helposti. On huomioitava, että kaikki kolme osaa ovat toiminnan kannalta yhtä tärkeitä, eikä järjestelmä voi toimia jos jokin näistä kolmesta peruspalasesta puuttuu.

KUVIO 1. Sähkösääennusteen tietojärjestelmän rakenne (Korhonen, P. 2012)

4.2 Tietokanta

Tietokanta on kokoelma yhteen liittyvää tietoa eli dataa. Se on loogisesti yhteenkuuluvien tallennettujen tietojen joukko, jota voidaan helposti käsitellä tietokantakielellä, kuten SQL. Tiedolla on tietokannassa aina jokin merkitys. Satunnaista valikoimaa tietoa ei pidetä tietokantana. Tietokanta on suunniteltu, rakennettu ja täytetty tiedolla jotakin tiettyä tarkoitusta varten. Sillä on myös jokin tarkoitettu käyttäjäryhmä ja joitain ennalta laadittuja ohjelmia joita palvelun loppukäyttäjät käyttävät. (Luimula 2011.)

Energiagurun tietokanta on relaatiotietokanta. Sillä tarkoitetaan että tietokanta koostuu tauluista, joihin tieto tallennetaan. Taulu on relaatiotietokannan peruselementti, ja muut tärkeät käsitteet ovat taulun lisäksi tietue ja kenttä. Tietueella tarkoitetaan taulun yhtä riviä, joka voi sisältää useita kenttiä. Kenttien nimet kuvaavat taulussa yhtä saraketta. Kenttä on siis taulussa yksi tieto. Tietokannan suunnittelussa tärkeimpiä asioita on se, että tiedon toistuvuus minimoidaan. Siitä käytetään nimitystä normalisointi.

TAULUKKO 1. Kuvitteellinen tietokannan taulu

Hetu	Etunimi	Sukunimi	Titteli
000000-0000	Olli	Opettaja	Lehtori
000001-0001	Ismo	Insinööri	Projekti-insinööri

Taulukko 1 voisi olla esimerkkitapaus tietokannan yhdestä taulusta. Taulun nimi on henkilo. Skandinaavisia merkkejä ei yleensä käytetä taulujen nimissä. Lisäksi taulut nimetään yleisesti yksikkömuotoon, vaikka taulu sisältääkin hyvin usein suuren määrän tietoa. Esimerkkitaulun ensimmäinen kenttä on henkilötunnus, ja se on samalla taulun pääavain. Pääavain on aina yksilöllinen, joten henkilötunnus voidaan valita pääavaimeksi johtuen siitä, että kahdella eri henkilöllä ei voi olla sama henkilötunnus. Esimerkkitaulussa on sekä etunimelle ja sukunimelle kentät erikseen, koska tietokannan suunnittelussa pyritään välttämään moniarvoisia sarakkeita. Viimeinen sarake on titteli, jonka avulla tietokannasta voitaisiin hakea esimerkiksi kaikki sellaiset henkilöt, jotka ovat koulutukseltaan insinöörejä.

Energiagurun tietokannan ohjelmisto on MySQL. MySQL on erittäin suosittu avoimeen lähdekoodiin perustuva tietokantaohjelmisto. Siitä on saatavana myös kaupallinen versio. MySQL:ää käytetään hyvin yleisesti Www-pohjaisissa sovelluksissa PHP:n kanssa juuri sen takia, että ne ovat yksityiselle käyttäjälle ilmaisia (Kolehmainen 2006, 281).

4.3 PHP -palvelu

Windows Phone -kehitysympäristössä ei ole mahdollista tehdä suoraan tietokantayhteyksiä. Siksi järjestelmään tehtiin PHP -kielellä skripti, joka hakee annettujen parametrien perusteella tietokannasta tietoa, joka voidaan vastaanottaa

pienoisohjelmaan. PHP -skripti saa pienoisohjelmalta parametreina kolme arvoa: valinta, lukumäärä ja tyyppi. Valinta on joko kotitalouspuoli, yrityspuoli tai yrityksen suojausennuste. Lukumäärä on haettavien tietojen määrä. Asetin maksimimääräksi 30, koska hyvin suurella määrällä haku hidastuisi ja kuvaajan päiväysakselista tulisi sekava. Tyyppi voi olla joko lyhyen tai pitkän aikavälin tarkastelu. Pitkä aikaväli on käytössä vain kotitalouspuolella.

Tein PHP -palveluun oman funktion kotitalouksille, yrityksille ja suojausennusteelle. Niitä kutsutaan parametrina saadun tyypin perusteella monivalintarakenteella. Kotitalouspuolella on lisäksi monivalintarakenne, jolla haetaan joko pitkän tai lyhyen aikavälin arvot. Pitkän aikavälin SQL -kyselyssä haetaan taulusta koko väliltä 30 arvoa sopivin välimatkoin. Hankaluuksia aiheutti se, että taulun id-numerointi ei ala numerosta yksi. Haettu tieto asetetaan mysql_fetch_array -toiminnolla taulukoiksi. Kyselyllä haetaan kaikki sopimustyyppien hinnat ja lisäksi niitä vastaavat päivämäärät. Kun tiedot on haettu, ne asettaan moniulotteiseen taulukkoon, jonka jälkeen tiedot on helppo tulostaa sisäkkäisellä for -toistorakenteella. Hintojen väleihin tulostetaan lisäksi erotinmerkki. Yksittäinen hinta erotellaan kaksoispisteellä ja hintasarjat putkitusmerkillä.

Yrityspuolen funktio on hyvin samankaltainen kuin kotitalouspuolellakin. Tieto kuitenkin haetaan eri taulusta ja tuotteet ovat kuukausituote, kvartaalituote ja vuosituote. Oman haasteensa toteutukseen toi se, että tiedot vaihtuvat päivämäärän perusteella. Yrityspuolen sähkösääennusteessa näytetään seuraavan kuukauden kuukausituote, seuraavan kvartaalin kvartaalituote ja seuraavan vuoden vuosituote. Kvartaali toteutettiin monivalintarakenteella ja merkkijonoja yhdistelemällä. Lähtökohtana oli aina tämänhetkinen päiväys, jonka perusteella oikean tuotteen hinnat ja päivämäärät haetaan kannasta.

Suojausennusteessa asiakas valitsee itselleen tuotteen, ja asettaa sille Energiagurun Www-palvelussa suojaushintapyynnön sekä hintaputken ylä- ja alarajan. Halutessaan asiakas voi sisällyttää suojaukseen aluekohtaisen lisän. PHP -palvelu hakee tietokantaan tallennetun tuotekoodin perusteella tuotteen sekä ylä- ja alarajat. Suojausennusteen funktio on myös hyvin samankaltainen kuin yrityspuolen sähkösääennuste. Kun tiedot on onnistuneesti haettu kannasta, funktio tulostaa arvot, jotka vastaanotetaan pienoisohjelmaan.

4.4 Pienoisohjelma

Pienoisohjelman tarkoituksena on havainnollistaa sähkön hintojen kehitys kuvaajan avulla. Ohjelman toteutus alkoi selvittämällä, onko WP7-ympäristöön olemassa valmiiksi kuvaajakomponentti. Kuvaajaa ei ollut saatavana vakiokontrollina, vaan sitä varten täytyi ladata ja asentaa Silverlight Toolkit For Windows Phone -työkalukirjasto. Se sisälsi kuvaajan ohella monia muita kontrolleja. Kuvaaja luotiin XAML -koodissa ja aluksi tavoitteena oli saada PHP -palvelu ja pienoisohjelma toimimaan yhdessä siten, että kuvaajaan saataisiin näkymään demonstroivasti satunnaiset arvotut liukuluvut. Jotta tiedot saisi haettua PHP- palvelusta, täytyi luoda WebClient -luokka. Sillä saadaan yhteys haluttuun palvelimeen, johon annettujen parametrien perusteella luokka lähettää pyynnön, ja PHP -palvelu vastaa pyyntöön arpomalla luvut ja näyttämällä ne asianmukaisesti eroteltuina.

Suurin haaste pienoisohjelmassa oli tiedon lukeminen ja käsittely. Vaati paljon aikaa ja yritystä, että tiedot saatiin luettua oikein. Kehitysvaiheessa käytin runsaasti try-catch -rakennetta, jolla virheiden selvittäminen oli helpompaa. Tiedoston lukeminen tapahtui StreamReader -luokalla. Luetun tietovirran rivi lisättiin listaan, jonka jälkeen alkoi tiedon käsittelyvaihe. Saatu data asetettiin

merkkijonotaulukoiksi ja eroteltiin asianmukaisesti. Sen jälkeen alustettiin moniulotteinen taulukko, johon kaikki hintatiedot asetettiin. Vasta tämän jälkeen luodaan listat oikeilla tietotyypeillä (*double*) ja merkkijonomuotoiset tiedot lisätään toistorakenteella listoihin muuttaen samalla tietotyyppi.

Kuvaajien tiedon varastoimiseen tarvittiin sanakirjarakennetta (*dictionary*). Se koostuu avaimesta ja arvosta. Avain on tietokannasta haettu päivämäärä ja sen tietotyyppi on merkkijono. Arvo on niin ikään tietokannasta saatu hintatieto, tietotyypiltään reaaliluku (*double*). Kun tiedot on lisätty sanakirjaan, ne lisätään XAML:lla luodun kuvaajan (*chart*) kuvaajasarjojen tietolähteeksi (*itemssource*). Tein C# -koodiin lisäksi toiminnon, joka selvittää saaduista hintatiedoista pienimmän ja suurimman. Pienimmästä arvosta vähennetään luku yksi, ja saatu arvo asetetaan kuvaajan hintaskaalauksen alarajaksi. Sama tehtiin ylärajan kanssa lisäämällä suurimpaan arvoon luku yksi.

Pienasiakaspuolella käyttäjä voi valita haettujen arvojen määrän luetteloruudusta (*listbox*). Vaihto huomioidaan välittömästi, eli tiedot ladataan aina uudestaan kun luetteloruudun valinta muuttuu. Näin käytettävyys on huomioitu, että ohjelman toiminta olisi mahdollisimman selkeää, nopeaa ja sujuvaa. Pienasiakas voi myös valita radionapilla haluaako hän pitkän vai lyhyen aikavälin tarkastelun. Pitkä aikaväli näyttää hinnat kolme vuotta taaksepäin. Lyhyellä aikavälillä asiakas voi esimerkiksi hakea kymmenen viimeisintä hintaa. Käyttäjä on huomioitu myös siten, että aina tietoja ladattaessa ohjelman yläreunassa näytetään Windows Phone 7:n latausanimaatio. Sillä indikoidaan käyttäjälle, että tietoja ladataan parhaillaan.

Yritysasiakkaiden sähkösääennuste näyttää kuvaajassaan aina seuraavan kuukauden, kvartaalin tai vuoden tuotteen. Tarkasteluväli on yhden kuukauden mittainen. Varmuuden vuoksi PHP -palvelussa haetaan arvoja nykyhetkestä kolme kuukautta taaksepäin. Tällä varmistetaan, että pienoisohjelma saa aina

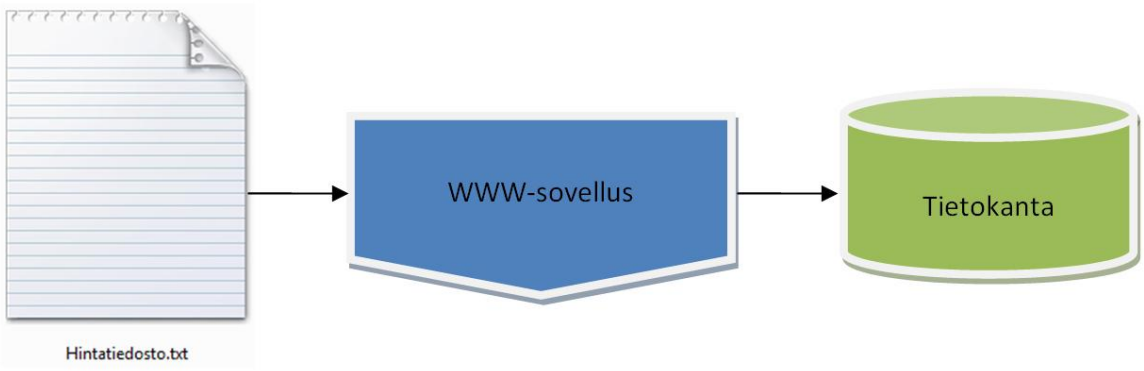
riittävän määrän arvoja. Halutessaan asiakas voi katsoa myös markkinakommentin. Se näytetään avautuvassa viesti-ikkunassa (*messagebox*).

Suojausennusteessa kuvaajaan piirretään asiakkaan tuote, hintaputken ylä- ja alarajat sekä suojaushintapyyntö. Valitettavasti Silverlight Toolkitin chart-komponentti ei tukenut rajoitusviivoja eikä tyhjiä pisteitä (*empty points*). Rajoitusviivat toteutettiin PHP -palvelussa kopioimalla kannasta saatua hintaa niin, että lukeminen voidaan tehdä pienoisohjelman C#-koodissa samalla tavalla kuin sähkösääennusteissakin. Suojausennuste vaatii vielä jatkokehitystä, sillä on alkeellinen tietoturvauhka lähettää asiakkaan käyttäjätunnus ja salasana verkon yli suojaamattomana. Kirjautumistietojen tallentamiseen käytettiin Isolated storagea, mikä parantaa käyttäjän käyttökokemusta.

Pienoisohjelman visuaalinen ilme rakentuu pitkälti perustavaa laatua olevan Windows Phone 7 -projektin ulkoasun pohjalta. Päänäkymä sisältää kuvaajan lisäksi sen yläpuolella Energiagurun liikemerkin sekä navigointilinkit (LIITE 1). Kuvaajan alapuolella on tapauskohtaisesti valintamahdollisuudet pitkän- ja lyhyen aikavälin tarkasteluun sekä haettavien tietojen määrään. Kuvaajan taustaväriksi valitsin vaaleansinisen, ja päänäkymän taustakuva on ruudukkotyyppinen mustanharmaa taustakuva. Kuvaajan läpinäkyvyyttä (*opacity*) on hieman vähennetty jotta kuvaaja sulautuu paremmin taustakuvaan. Kun puhelin käännetään vaaka-asentoon (*landscape*), kuvaaja suurennetaan koko näytön kokoiseksi. Yritysasiakkaiden sähkösääennusteen päänäkymä on vastaavanlainen kotitalousasiakkaiden kanssa (LIITE 2).

5 HINTAPÄIVITYKSEN AUTOMATISOINTI

Energiaguru saa kuukausittain Suomen Energiamarkkinavirastolta tekstimuotoisen hintatiedoston, joka sisältää sähköyhtiöiden tariffien hinnat. Tiedosto sisältää hinnat niin yleis-, aika- kuin kausituotteiden myyntiin. Tehtävänäni oli rakentaa Www-sovellus, jolla uudet hinnat päivitetään tietokantaan. Tavoitteena oli säästää aikaa ja vaivaa, koska aiemmin hinnat lisättiin kantaan käsin.



KUVIO 2. Hintapäivityksen tietojärjestelmän rakenne (Korhonen, P. 2012)

Hintapäivityssovellus jakaantuu karkeasti kolmeen vaiheeseen. Ensimmäisenä vaiheena on tiedoston lukeminen, toisena tiedon käsittely ja viimeisenä tiedon lisääminen tietokantaan.

5.1 Tiedoston lukeminen

Tein yleis-, aika- ja kausimyyntitariffeille jokaiselle oman funktion. Kun päivittäjä aloittaa päivityksen, hän valitsee valintalomakkeelta luettavan tiedoston omalta tietokoneeltaan. Tähän on käytetty PHP:n esimääriteltyä $_FILES -muuttujaa.

Päivittäjä valitsee samalla osion ja kuukauden, joiden tiedot halutaan päivittää. Koko tiedoston sisältö luetaan taulukoksi (*array*) PHP:n file -funktiolla, minkä jälkeen funktio palauttaa luetun tiedoston käsittelyä varten. Tiedoston lukemisen funktiota kutsutaan jokaisen osion funktion alussa. Tiedoston käsittelyä varten täytyi varmistaa, että jokaisessa ohjelman vaiheessa käytetään samaa merkistökoodausta. Siksi hintatiedoston merkistä täytyy muuttaa ennen päivittämistä ja käsittelyä UTF-8 -muotoon. Muuten ohjelma voi antaa odottamattomia tuloksia.

5.2 Tiedon käsittely

Hintatiedostossa on säännöllinen rakenne, jonka perusteella tiedot voidaan käsitellä. Rakenne koostuu pääosin puolipisteistä, joilla erotellaan hinnat ja muut tiedot toisistaan sekä tehdään rivin tunnistus mahdolliseksi. Yhtiön nimen perässä on aina säännöllinen määrä puolipisteitä, joten sen avulla sähköyhtiö voidaan tunnistaa. Yhtiön nimen jälkeen seuraavalla rivillä on tariffi ja sen hinnat. Hinnat koostuvat perusmaksusta eri sulakekokoille ja osiosta riippuen sähkön siirtohinnasta. Hintatiedosto sisältää myös muutakin tietoa hintojen lisäksi kuten hintajakson alkuaika ja loppuaika. Lisäksi uusimmissa hintatiedostossa on kerrottu myös sopimustyyppi, mikä helpottaa päivittämistä paljon.

Tiedon käsittelyssä tarvittiin erityisesti PHP:n merkkijonofunktioita. Explode -funktiolla voidaan erottaa kukin yksittäinen tieto toisistaan. Str_replace -funktiolla voidaan korvata merkkijono toisella. Esimerkkitapauksena yhtiön nimen perässä olevat puolipisteet korvataan tyhjällä merkkijonolla. Toistaiseksi tuntemattomasta syystä tyhjä merkkijono lisäsi yhtiön nimeen rivinvaihdon, joka aiheutti ongelmia tietokannassa. Siksi myös rivinvaihto täytyi poistaa erikseen.

Joissakin tilanteissa hintatiedosto sisälsi poikkeuksellisia rivejä varsinkin kausimyynnissä. Tietyn ajanjakson tarjoukset vaikeuttivat ohjelman toteutusta. Silloin listauksessa rivejä oli joko ylimäärä tai normaalitapausta vähemmän. Näitä poikkeustilanteita varten täytyi luoda säännöt, miten poikkeuksia käsitellään. Silloin tiedon luvun silmukkarakenteen indeksiä joko kasvatettiin tai vähennettiin. Tämä ratkaisu vaatisi vielä kehitystä, sillä vastaan voi tulla tilanne jota sovellus ei osaa käsitellä ja antaa siten virheellistä tietoa.

5.3 Tiedon lisääminen tietokantaan

Kun tieto on saatu luettua ja käsiteltyä, se voidaan kirjoittaa tietokantaan. Tieto luetaan ja käsitellään toistorakenteessa, jonka lopussa käsitelty tieto lisätään tietokantaan PHP:hen upotetulla SQL:n INSERT INTO -lauseella. Aluksi hintapäivitys oli tarkoitus toteuttaa niin, että tieto lisätään kantaan tai olemassa olevat tietueet päivitetään UPDATE -lauseella. Kiireestä johtuen hintapäivitys toteutettiin kuitenkin niin, että vanhat tiedot poistetaan kokonaan tietokannasta, ja uudet tiedot lisätään yhdellä kertaa tietokantaan. Se miksi alussa oltaisiin haluttu käyttää UPDATE- ja INSERT INTO -lauseita rinnakkain oli se, että tieto luettiin Energiagurun palvelussa id-numeron perusteella. Silloin olisi vaadittu aina se, että yksittäisellä tariffilla on sama id-numero jokaisen kuukauden taulussa. Onneksi tästä kuitenkin ainakin toistaiseksi luovuttiin, koska päivitys olisi ollut siinä tapauksessa melko työläs tehtävä. Hintapäivityssovellus olisi päivittänyt olemassa olevat samat tiedot UPDATE:lla, mutta loput tariffit olisi joutunut käsin osoittamaan, mihin tariffiin tieto kirjoitetaan. Kokonaan uudet tariffit olisi kirjoitettu normaalisti INSERT INTO -lauseella.

5.4 Lisätoiminnot

Järjestelmän toiminnan kannalta on oleellista, että jokaiselle tariffille on kerrottu sopimustyyppi. Vanhan mallisissa hintalistauksissa sopimustyypin selvittäminen oli melkoista salapoliisin työtä, koska sitä ei aina kerrottu ja jos kerrottiin, tapoja oli monia erilaisia. Siksi hintapäivityssovellukseen täytyi kehittää lisätoiminto, jolla sopimustyypit saatiin selvitettyä ja lisättyä tietokantaan. Aluksi rakensin muokkaustoiminnon, jolla jokainen sopimustyyppi täytyi lisätä yksitellen muokkaussivulla. Samalla pystyi korjaamaan kirjoitusvirheitä. Oli kuitenkin sanomattakin selvää, että sopimustyypin lisäämiseen tarvittiin nopeampi ja parempi ratkaisu. Lopulta muokkaussivu päädyttiin toteuttamaan siten, että sopimustyypit lisätään kukin erikseen tekstikenttään, mutta jokainen tieto tallennetaan tietokantaan yhdellä napin painalluksella. Muokkaussivustolla tarvittiin SQL:n SELECT -lauseketta tiedon hakemiseen ja UPDATE -lauseketta tietojen päivittämiseen.

Yksittäinen tietue voidaan myös poistaa päivityksen jälkeen tietokannasta, jos hintalistauksessa ilmenee kaksi samanlaista tariffia. Muita erityistoimintoja hintapäivityssovelluksessa ei ole tietoturvaratkaisujen lisäksi. Päivityssovellus kysyy valintalomakkeelle siirryttäessä aina käyttäjätunnuksen ja salasanan. SQL-hyökkäykset ovat kanssa huomioitu lisäämällä muuttujiin turvallisuusmäärittelyitä jotka ainakin joiltakin osin estävät ilkivallan teon tietokantaan.

6 TULOKSET JA POHDINTA

Onnistuin opinnäytetyössäni kohtalaisesti. Hintapäivityksen jotkut tekniset ratkaisut olisivat pitäneet olla ehdottomasti toisenlaisia, jotta sovelluksen jatkokehittäminen olisi helpompaa ja vaivattomampaa. Tulevaisuudessa tietokantaa uudistetaan paremmaksi, ja tällöin myös hintapäivityssovellus vaatii muutoksia. Pidin kuitenkin alusta asti tärkeimpänä sitä, että ohjelma kuitenkin toimii kuten pitääkin. Hintapäivitys haluttiin toimintakuntoon mahdollisimman nopeasti, joten saavutettu nopeus vaati mutkien oikomista teknisessä toteuttamisessa. Kuitenkin voin olla jonkun verran tyytyväinen, että sain työelämäohjaajaltani useaan otteeseen positiivista palautetta päivittämisen sujuvuudesta.

Sähkösääennuste pienoisohjelmana onnistui tekniseltä toteutukseltaan paremmin. Ainoa kysymys joka jäi oli se, onko mobiiliversio riittävän informatiivinen ja mielenkiintoinen ajatellen esimerkiksi yksittäistä kotitalouskäyttäjää. Käyttäjä näkee kuvaajasta hintojen kehityksen ja lisäksi halutessaan markkinakommentin, mutta pitäisikö ohjelman antaa vielä enemmän jotakin tietoa? Työ kuitenkin vaatii joka tapauksessa jatkokehittelyä, koska sähkösääennustetta ei ole vielä julkaistu virallisesti Nokian kauppapaikkaan.

Kokonaisuutena opinnäytetyö oli pitkä ja haastava projekti. Vaikeinta oli jokaisessa vaiheessa alkuun pääseminen. Kun on tilanne ettei osaa hahmottaa sovelluksen toimintaa edes niin sanottuna pseudokoodina, on vaikea lähteä rakentamaan osia sovellukseen tyhjästä. Sitten kun pääsi juonesta kiinni, oli opinnäytetyön tekeminen myös palkitsevaa. Kun sai yhden osakokonaisuuden toimimaan, sen voi tavallaan unohtaa kokonaan ja keskittyä seuraavaan haasteeseen.

LÄHTEET

Adage Oy. 2011. Älypuhelimet syrjäyttämässä tietokoneen. Www-dokumentti. Saatavissa: http://adage.fi/. Luettu 29.9.2012.

Hangasmaa, Heikki. 2010. Käytettävyyden merkitys ja käytettävyystestaus. Opinnäytetyö. Vaasan ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma.

Juola, Toni. 2011. Oulun eteläisen alueen julkisten toimijoiden kansainvälisten yhteyksien karttapalvelu. Opinnäytetyö. Keski-Pohjanmaan ammattikorkeakoulu. Mediatekniikan koulutusohjelma.

Kolehmainen, Kauko. 2006. Php & MySQL - Teoriasta käytäntöön. Helsinki: readme.fi.

Kolari, Mika. 2012a. C# perusteet. Esimerkkiohjelma. Www-dokumentti. Saatavissa: http://mikakolari.fi/csharp-dotnet/perusteet/ohjelman-rakenne/. Luettu 8.12.2012.

Kolari, Mika. 2012b. C#/.NET luokkakirjastoja. List<T>. Www-dokumentti. Saatavissa: http://mikakolari.fi/csharp-dotnet/net-luokkakirjastoja/kokoelmat/list/. Luettu 8.12.2012.

Kollanus, Sami. 2012. Johdatus PHP-kieleen. Www-dokumentti. Saatavissa: http://users.jyu.fi/~kolli/ITK215_05/php/?sivu=operaattorit. Luettu 4.7.2012.

Korpela Jukka K. & Linjama Tero. 2005. Web-suunnittelu. Jyväskylä: Docendo Finland Oy.

Leander, Suvi. 2006. Matkapuhelimien käytettävyys. Seminaariesitelmä. Www-dokumentti. Saatavissa: http://www.sis.uta.fi/~pi52316/vtsem/suvi_leander/seminaariesitys2.pdf. Luettu 10.12.2012.

Luimula, Mika. 2011. SQL -ja relaatiotietokanta. Luentomuistiinpanot. Keski-Pohjanmaan ammattikorkeakoulu. Ylivieskan yksikkö.

Microsoft. 2012. MSDN. Visual C#. Www-dokumentti. Saatavissa: http://msdn.microsoft.com/en-us/library/kx37x362.aspx. Luettu 7.12.2012.

Petzold, Charles. 2010. Programming Windows Phone 7. Washington: Microsoft Press.

Qvik Oy. 2011. Mobiilikehitys. Www-dokumentti. Saatavissa: http://mobiilikehitys.fi/2011/10/. Luettu 19.7.2012.

Sanastokeskus TSK ry. 2012. Hakemistot. Www-dokumentti. Saatavissa: http://www.tsk.fi/tsk/termitalkoot/hakemistot-267.html?page=get_id&id=ID193&vocabulary_code=TSKTT. Luettu 17.7.2012.

Sinkkonen Irmeli, Kuoppala Hannu, Parkkinen Jarmo & Vastamäki Raino. 2006. Käytettävyyden psykologia. 3., uudistettu painos. Helsinki: Edita Publishing Oy.

The PHP Group. 2012a. PHP Introduction. Www-dokumentti. Saatavissa: http://fi2.php.net/manual/en/intro-whatis.php. Luettu 3.7.2012.

The PHP Group. 2012b. PDO Introduction. Www-dokumentti. Saatavissa: http://www.php.net/manual/en/intro.pdo.php. Luettu 18.11.2012.

LIITE 1


SÄHKÖSÄÄENNUSTE KOTITALOUKSILLE
EnergiaGuru
Ennuste yrityksille
Näytä markkinakommentti
Ennuste kotitalouksille
snt/kWh
12
10
8
6
4
2
Nordpool
Toim. Velv.
Toist. Voim.
Tarjous 1v.
Tarjous 2v.
Pitkän aikavälin tarkastelu
Viimeisimmät arvot

LIITE 2


SÄHKÖSÄÄENNUSTE YRITYKSILLE
EnergiaGuru
ENERGIA Guru
Ennuste kotitalouksille
Näytä markkinakommentti
Ennuste yrityksille
€/MWh
44
42
40
38
Kvartaali
NPENOQ113
Vuosi
NPENOYR13
Lähin kuukausi
NPENOMJAN13
Siirry suojausennusteeseen